Haawking_DSC28027/34 数字信号控制器

迁移指南

编号：

北京中科昊芯科技有限公司

2022 年 3 月

V1.4

历史版本记录

版本号	时间	内容描述	审核人	批准人
V0.1	2021.2.26	初版，指南框架		
V0.2	2021.03.02	增加 PWM、ADC 模块指导		
V0.3	2021.03.11	增加 XINT 模块指导		
V0.4	2021.03.25	增加编程模式差异相关章节		
V0.5	2021.08.04	根据量产版芯片修改		
V0.6	2021.08.11	增加嵌套中断的说明		
V0.7	2021.08.12	内部时钟说明		
V0.8	2021.09.08	更改 SEGGER 实时运行库、驱动库更新和修改等		
V0.9	2021.09.16	增加关于 AIODIR 的使用说明		
V1.0	2021.11.17	增加了多个差异寄存器说明		
V1.1	2021.11.24	更新死区的延时 Half 使用说明		
V1.2	2022.01.24	更新修改中断及中断嵌套		
V1.3	2022.02.10	增加 sizeof(),memcpy(),typedef enum 使用		
V1.4	2022.03.18	增加 IDE 支持源文件的类型		

目录

1.芯片与友商差异

本版本指南适用于 Haawking-HX2000 系列芯片中的 DSC28027 和 DSC28034 型号。

1.1 CPU 核

Haawking HX2000 系列芯片基于开放指令集架构 RISC-V 开发，支持 RV32IMC 标准指令集+DSP 自定义指令集，在程序开发的时候，高级语言开发的程序代码由编译器负责生成可执行文件，汇编语言开发的代码，则需要手动根据 RISC-V 指令集进行修改。

1.2 PWM 模块

1.2.1 CLKDIV

该模块比友商增加了 CLKDIV 寄存器，因为我们的主频是友商的 2 倍，所以如果客户需要，需要先分频一次；详细信息请参阅我们用户手册。

1.2.2 死区的延时 Half 使用说明

友商：DBCTL.HALFCYCLE 位=0：死区的延时采用 TBCLK 时钟；

DBCTL.HALFCYCLE 位=1：死区的延时采用 2 倍 TBCLK 时钟；

HX：新增 DBCTL.HALFFEN 位，DIVCLK 寄存器。

死区的延时采用的时钟和与 DBCTL.HALFCYCLE、DBCTL.HALFFEN、CLKDIV 寄存器、TBCTL.CLKDIV、TBCTL.HSPCLKDIV 五项有关，其中 TBCTL.CLKDIV 、TBCTL.HSPCLKDIV 对死区的延时影响只有分频和不分频的区分。

①在 TBCTL.CLKDIV、TBCTL.HSPCLKDIV 同时为 0，无论 CLKDIV 等于 0 还是 1 的情况下，

PWM 的死区的延时采用系统都是和 DBCTL.HALFFEN 有关，此时用法和友商的用法不同。

用法如下：

HALFCYCLE 位无用，配置无效；

HALFEN 位=0：死区采用 2 倍 TBCLK 时钟；

HALFEN 位=1：死区采用 TBCLK 时钟；

②其他情况与友商用法相同

配置真值表如下表所示：

序号	DBCTL.HALFCYCLE	DBCTL.HALFFEN	CLKDIV 寄存器	TBCTL.CLKDIV	TBCTL.HSPCLKDIV	DBRED	实际的 DBRED
1	0	0	1(2 分频)	011（8 分频）	011（6 分频）	0x10	0x10

2	0	1	1(2 分频)	011（8 分频）	011（6 分频）	0x10	0x10
3	1	0	1(2 分频)	011（8 分频）	011（6 分频）	0x10	0x8
4	1	1	1(2 分频)	011（8 分频）	011（6 分频）	0x10	0x8
5	0	0	0(1 分频)	011（8 分频）	011（6 分频）	0x10	0x10
6	0	1	0(1 分频)	011（8 分频）	011（6 分频）	0x10	0x10
7	1	0	0(1 分频)	011（8 分频）	011（6 分频）	0x10	0x8
8	1	1	0(1 分频)	011（8 分频）	011（6 分频）	0x10	0x8
9	0	0	1(2 分频)	000(1 分频)	000（1 分频）	0x10	0x8
10	0	1	1(2 分频)	000(1 分频)	000（1 分频）	0x10	0x10
11	1	0	1(2 分频)	000(1 分频)	000（1 分频）	0x10	0x8
12	1	1	1(2 分频)	000(1 分频)	000（1 分频）	0x10	0x10
13	0	0	0(1 分频)	000(1 分频)	000（1 分频）	0x10	0x8
14	0	1	0(1 分频)	000(1 分频)	000（1 分频）	0x10	0x10
15	1	0	0(1 分频)	000(1 分频)	000（1 分频）	0x10	0x8
16	1	1	0(1 分频)	000(1 分频)	000（1 分频）	0x10	0x10

1.3 ADC 模块

1.3.1 ADC 模块分频

ADC 模块的典型工作频率为 25MHz，最高工作频率为 40MHz，所以在芯片工作在 120MHz 的时候，需要设置为 4 分频，即

```
1.     AdcRegs.ADCCTL2.bit.CLKDIV2EN = 1; //sysclk/4
2.     AdcRegs.ADCCTL2.bit.CLKDIV2EN = 0; //sysclk/2
```

否则，ADC 有效位数和性能将会受到影响。

1.3.2 增加以下位定义

ADCCTL1.ADCRDY：ADC 上电完成标志位，0=未完成；1=上电完成
ADCCTL2.SELDO_HS_LU：ADC 结果符号位，0=无符号数；1=有符号数；
ADCCTL2.EN_ALG_MEAN：温度传感器均值算法使能位，0=关闭，1=使能；
ADCCTL2.ALG_MEAN：均值算法采样点选择，deffault=10d
ADCCTL2.ADJ_TD_GA：Gain trim，deffault=1000；
ADCCTL2.ADJ_TD_OS：Offset trim，deffault=1000；

不支持外部电压基准，VERFLO 和 VERFHI 无用；

1.3.3 ADCREFTRIM 和 ADCOFFTRIM 寄存器未使用

1.3.4 关于 AIODIR 的使用

AIODIR 寄存器不支持位域操作，只支持 32 位寄存器写，且读返回无效值（只适用当前版本）.

```
1.      GpioCtrlRegs.AIODIR.all = 0x0000;//输入
2.      GpioCtrlRegs.AIODIR.all = 0xFFFF;//输出
```

1.4 CAN 模块

寄存器与友商一致，可以直接代码替换。

1.5 LIN 模块

寄存器与友商一致，可以直接代码替换。

1.6 SCI 模块

寄存器与友商一致，可以直接代码替换。

1.7 IIC 模块

寄存器与友商一致，可以直接代码替换。

1.8 比较器模块

相比友商增加了增加 DACEX 寄存器，详细信息请参阅我们的用户手册。

1.9 内部 OSC 模块

中科昊芯 DSC28027/34 内部集成两个 12MHz 晶振，不同于友商的 10MHz，例如：友商的 4 倍频的时候是 40MHz，中科昊芯 DSC28027/34 的就是 48MHz，友商的 10 倍频的时候是 100MHz，中科昊芯 DSC28027/34 的就是 120MHz；使用的时候，一定注意！

在 SYSCTL 模块中，PLLSTS、PLLCR、LOSPCP、BORCFG、DEVICECNF、REVID、LDOTRIM、CLASSID，标志位和友商不同，具体信息请参阅我们的用户手册。

1.10 SPI 模块

寄存器与友商基本一致，可以直接代码替换。

1.10.1 SPI 模块 CS 信号翻转功能

我们现在 DSC28027 有 CS 翻转，友商的 28027 没有。

1.11 eQEP 模块

寄存器与友商一致，可以直接代码替换。

1.12 eCAP 模块

寄存器与友商一致，可以直接代码替换。

1.13 HRPWM 模块

HRPWR 寄存器和友商的有一些区别，具体信息请参阅我们的用户手册；

1.14 HRCAP 模块

比友商多出 HCCAL，HCCALMEP，HCMEPSTATUS 这三个寄存器，详细说明请参阅我们的用户手册。

1.15 外部中断 XINT

寄存器与友商一致，可以直接代码替换。

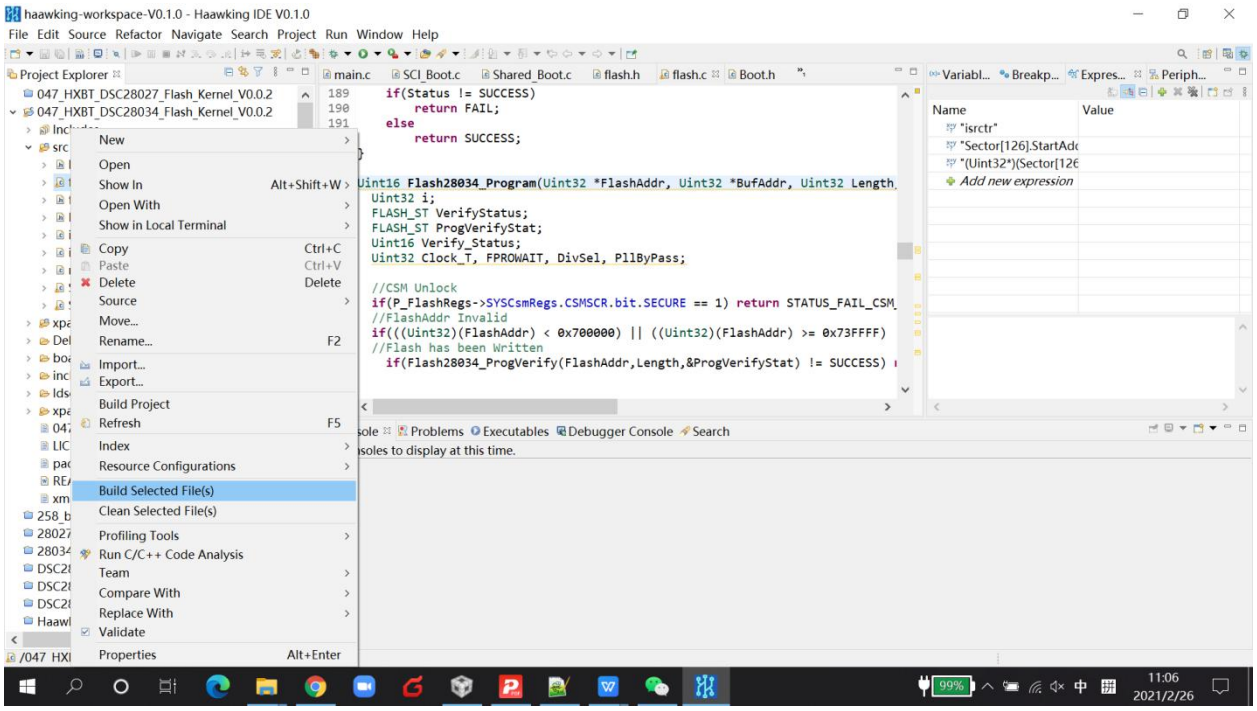
2.开发工具差异

Haawking HX2000 系列芯片采用中科昊芯自主研发的开发工具，当前版本使用习惯与友商 CCS5 之后的版本一致，包括常用的 CCS6 和 CCS10。

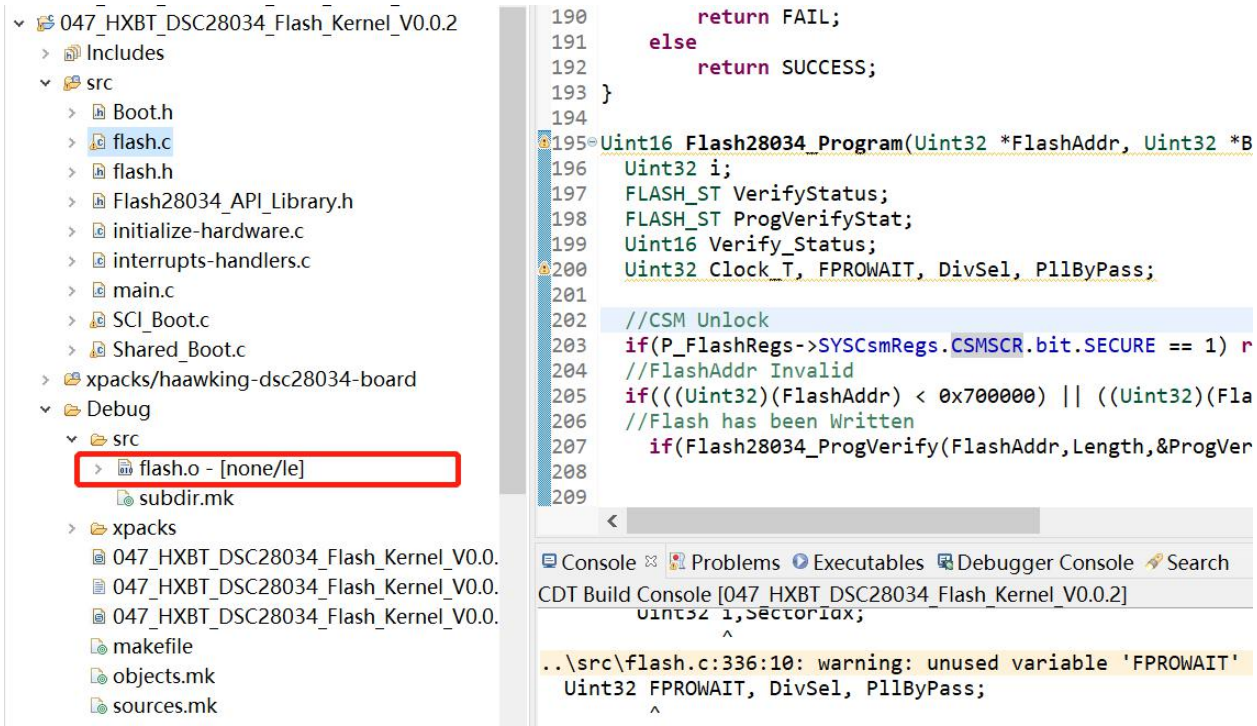
2.1 编译单个源文件

Haawking IDE 支持单个源程序文件的编译，可以加速开发过程。

在 Haawking IDE 的项目索引栏中，右键选中单独编译的文件，比如 flash.c，在菜单中选择 Build Selected File(s)，就可以只编译 flash.c 文件。



编译完成以后，.o 文件在 Debug 目录下。



2.2 CodeStartBranch

友商提供的 CodeStartBranch.asm，该文件是 DSP 程序的入口文件，C 语言编程的时候，该文件能够执行 MAIN 函数前的操作，主要包括选择芯片 C27 或者 C28 地址模式，初始化相关存储器和缓冲区。

我们的开发工具中去掉了该文件，改为 common/syscall.c，完成同样的初始化功能。

同时，对于部分需要在 RAM 中运行的程序，common/syscall.c 中会自动将代码从 Flash 中拷贝到 RAM 中，不需要用户手动拷贝。

2.3 CSMPasswords.asm

友商提供的 CSMPasswords.asm 文件，用户可以用来修改 Flash 中密码区。

```
37      .sect "csmpasswds"
38
39      .int  0xFFFF      ;PWL0 (LSW of 128-bit password)
40      .int  0xFFFF      ;PWL1
41      .int  0xFFFF      ;PWL2
42      .int  0xFFFF      ;PWL3
43      .int  0xFFFF      ;PWL4
44      .int  0xFFFF      ;PWL5
45      .int  0xFFFF      ;PWL6
46      .int  0xFFFF      ;PWL7 (MSW of 128-bit password)
```

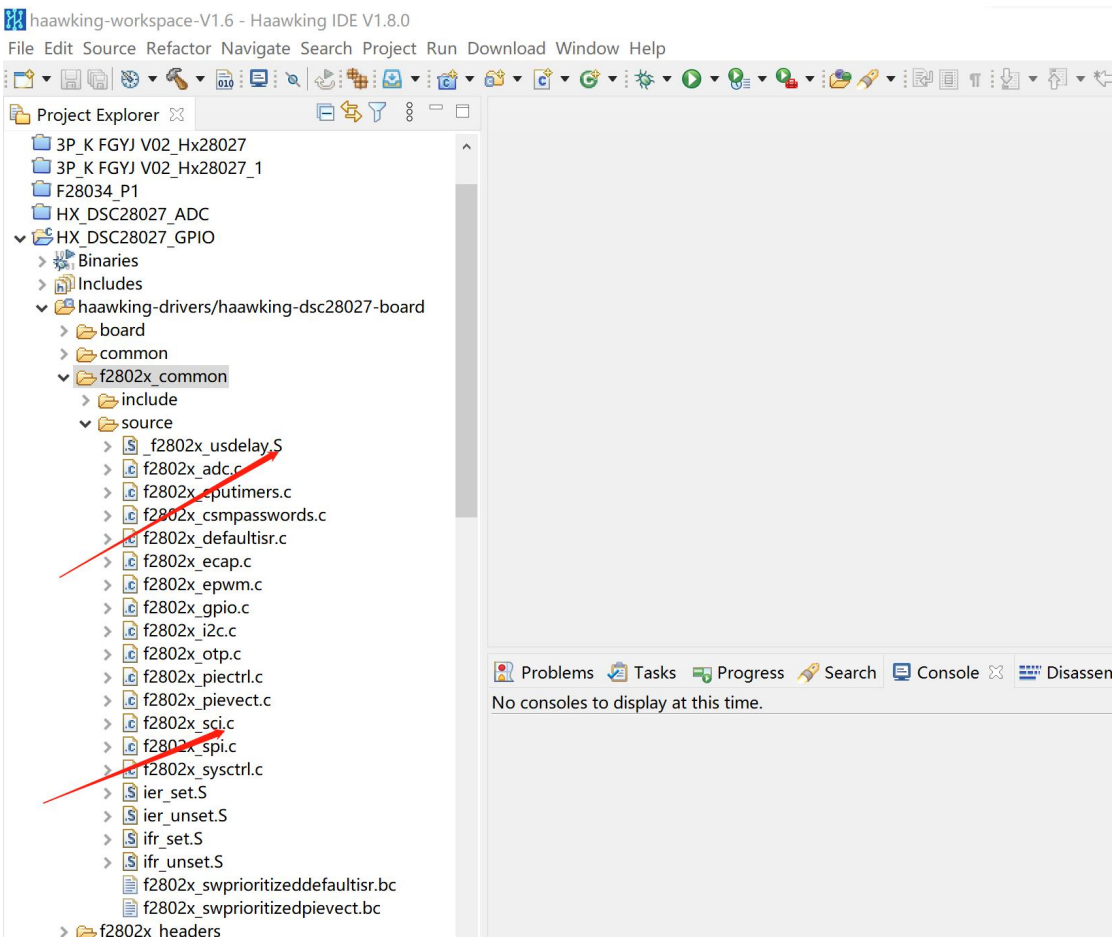
中科昊芯的 CSM 模块与友商一样，也是 128 位加密，不同之处在于中科昊芯是 4 个 32 位的密码。在中科昊芯提供的“f2802x_csmpasswords.c”文件里。

```
3 /*************************************************
4 password file
5 *************************************************************/
6
7 volatile struct CSM_PWL  CODE_SECTION(".CsmPwl") CsmPwl =
8 {
9     0xFFFFFFFF,
10    0xFFFFFFFF,
11    0xFFFFFFFF,
12    0xFFFFFFFF,
13 };
```

2.4 IDE 支持的 C 和汇编源文件的后缀名

对于 Haawking IDE V1.8.1 及之前版本,汇编文件的后缀名必须是大写的.S 格式，例如 crt.S; C 的源文件的后缀名必须是小写的.c；例如 Main.c;

所以在移植的时候，如果看到友商移植过来的文件名称里有.C 的时候，一定要改成.c。如下图：Haawking IDE 支持如下：


haawking-workspace-V1.6 - Haawking IDE V1.8.0
File Edit Source Refactor Navigate Search Project Run Download Window Help
Project Explorer
3P_K FGYJ V02_Hx28027
3P_K FGYJ V02_Hx28027_1
F28034_P1
HX_DSC28027_ADC
HX_DSC28027_GPIO
Binaries
Includes
haawking-drivers/haawking-dsc28027-board
board
common
f2802x_common
include
source
_f2802x_usdelay.S
f2802x_adc.c
f2802x_eputimers.c
f2802x_csmpasswords.c
f2802x_defaultisr.c
f2802x_ecap.c
f2802x_epwm.c
f2802x_gpio.c
f2802x_i2c.c
f2802x_otp.c
f2802x_piectrl.c
f2802x_pievect.c
f2802x_sci.c
f2802x_spi.c
f2802x_sysctrl.c
ier_set.S
ier_unset.S
ifr_set.S
ifr_unset.S
f2802x_swprioritizeddefaultisr.bc
f2802x_swprioritizedpievect.bc
f2802x_headers
Problems
Tasks
Progress
Search
Console
Disassen
No consoles to display at this time.

3. 编程模式差异

中科昊芯投入很多资源和精力，为客户提供了和友商一样的位域驱动文件，在编程的时候，可以完全跟友商一样。

```
64
65    InitPieVectTable();
66
67    EALLOW;  // This is needed to write to EALLOW protected registers
68    PieVectTable.ADCINT1 = &adc_isr;
69    PieVectTable.ECAP1_INT = &ecap1_isr;
70    PieVectTable.SCIRXINTA = &sciaRxFifoIsr;
71    PieVectTable.XINT1 = &Drxint1_isr;
72    EDIS;    // This is needed to disable write to EALLOW protected registers
73
74    EALLOW;
75    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
76    EDIS;
77
78 ;//  InitEPwm1Example();
79
80
81    EALLOW;
82    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;
83    EDIS;
```

3.1 CMD 文件

中科昊芯提供链接文件时 LD 格式，遵循 GNU LD 标准；同时，根据友商 2803X 芯片的内存映射进行修改。

```
18 MEMORY
19 {
20   /*PAGE 0:*/
21   PRAMM0 (rwx)   : ORIGIN = 0x0       , LENGTH = 0x800   /* RAM M0*/
22   PRAMM1 (rwx)   : ORIGIN = 0x800     , LENGTH = 0x800   /* RAM M1*/
23
24   AdcRegs_FILE (rw)   :  ORIGIN  = 0x00001400 , LENGTH = 156
25
26   AdcResult_FILE (rw)   :  ORIGIN  = 0x0000149c , LENGTH = 64
27
28   CpuTimer0Regs_FILE  (rw) : ORIGIN = 0x00001800 ,  LENGTH = 16
29   CpuTimer1Regs_FILE  (rw) : ORIGIN = 0x00001810 ,  LENGTH = 16
30   CpuTimer2Regs_FILE  (rw) : ORIGIN = 0x00001820 ,  LENGTH = 16
31
32   PieCtrlRegs_FILE(rw)   : ORIGIN  = 0x00001900 , LENGTH = 104
33
34   XIntruptRegs_FILE (rw) :   ORIGIN  = 0x00001968 , LENGTH = 124
35
36   PieVectTable_FILE (rw) :   ORIGIN  = 0x00001A00 , LENGTH = 448
37
38   PieEmuRegs_FILE (rw)    :  ORIGIN = 0x1BC0, LENGTH = 4
39
40   DmaRegs_FILE (rw)       :  ORIGIN  = 0x00001C00 , LENGTH = 1024
41
```


段设置，可以将不同的代码段，放到不同的存储分区。NOLOAD 属性和 LOAD 属性使用，与友商保持一致。

```
100
101 SECTIONS
102 {
103
104   .AdcRegs(NOLOAD) :  { *(.AdcRegs) }  >  AdcRegs_FILE
105
106   .AdcResult(NOLOAD) : { *(.AdcResult)  }  > AdcResult_FILE
107
108    .ECap1Regs(NOLOAD) : { *(.ECap1Regs) }  > ECap1Regs_FILE
109
110   .CpuTimer0Regs(NOLOAD) : { *(.CpuTimer0Regs) }     >  CpuTimer0Regs_FILE
111   .CpuTimer1Regs(NOLOAD) : { *(.CpuTimer1Regs) }     >  CpuTimer1Regs_FILE
112   .CpuTimer2Regs(NOLOAD) : { *(.CpuTimer2Regs) }     >  CpuTimer2Regs_FILE
113
114
115   .PieCtrlRegs(NOLOAD) : { *(.PieCtrlRegs)  }  > PieCtrlRegs_FILE
116
117   .PieVectTable(NOLOAD) : { *(.PieVectTable)  } >  PieVectTable_FILE
118
119   .XIntruptRegs(NOLOAD) :  { *(.XIntruptRegs) }  > XIntruptRegs_FILE
120
121   .PieEmuRegs(NOLOAD)    :  { *(.PieEmuRegs) } > PieEmuRegs_FILE
122
123   .EQep1Regs(NOLOAD)  : { *(.EQep1Regs) }  >  EQep1Regs_FILE
124
125   .SpiaRegs(NOLOAD) : { *(.SpiaRegs) } > SpiaRegs_FILE
```

段名字　段属性　段所在区

3.2 RAM 运行

友商的 2000 系列编译器支持 Progma 语法，告诉编译器如果修改一个特定函数、目标文件或者一段代码的属性，例如 CODE_SECTION，就是为某一个函数分配 Section，使用方式如下：

```
#pragma CODE_SECTION(funcA, "codeA")
```

HX2000 为了实现同样的功能，通过定义一个宏，如下：

```
#define CODE_SECTION(v) __attribute__((section(v)))
```

其中，v 表示将某一函数或者全局变量指定到的 SECTION 名。使用的时候，可以参考如下格式：

```
volatile int array[10]CODE_SECTION(“dmaisr”);
void dma_isr()CODE_SECTION(“dmaisr”){    }
```

只需要在函数和变量定义的时候进行属性指定即可，如，想将函数 dma_isr 放在 L0 中运行，则可以使用如下方式完成：

```
void dma_isr()CODE_SECTION(“L0”){    }
```

或者

```
void CODE_SECTION(“L0”) dma_isr(){    }
```

CODE_SECTION 宏已经在驱动中定义，用户可以直接使用。

3.3 中断服务程序

友商的编译器支持 interrupt 或者__interrupt 关键词，将中断服务程序映射到中断向量表，分配中断服务程序的地址。

```
interrupt void  adc_isr(void)
```

Haawking IDE 也提供了实现同样功能的宏定义，使用方式如下：

```
INTERRUPT void  adc_isr(void)
```

如果中断服务程序需要在 ram 中运行，则可以与 3.2 节同时使用：

```
INTERRUPT void CODE_SECTION("L0") adc_isr(void)
```

昊芯的芯片在使用嵌套中断的时候和友商大部分是一样的，区别在于下面的红色部分，我们需要在中断函数的开头和结尾增加一个保存状态寄存器的宏，其他的使用方式相同。如下：

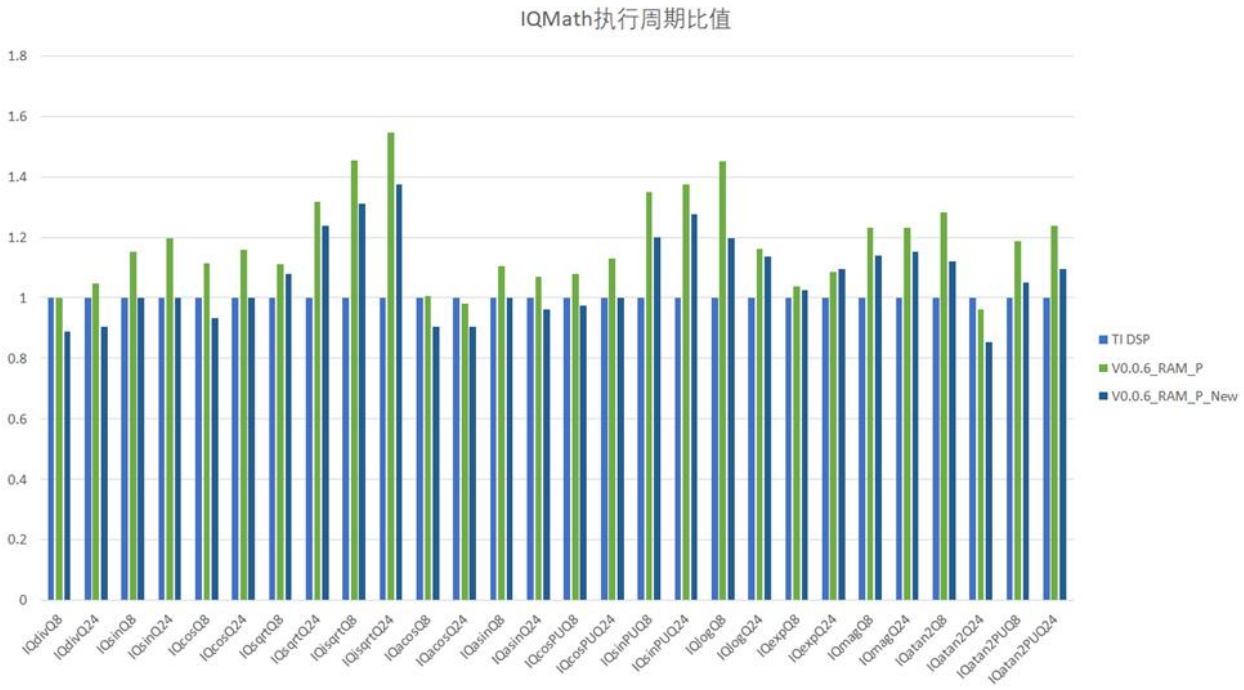
```
1.    INTERRUPT void INT13_ISR(void)
2.    { //保存返回 main 函数的地址以及其它状态，必须在 EINT 前执行
3.    _SAVE_IRQ_STATUS();
4.      Uint16_t  TemPIEIER;
5.     //保存 PIEIER 寄存器数值
6.      Timer0PIEIER = PieCtrlRegs.PIEIER1.all;
7.     // 在 IER 寄存器层级，使能嵌套中断 XINT1 所在组的中断
8.      IER |= M_INT1;
9.     // 在 IER 寄存器层级，屏蔽嵌套中断 XINT1 所在组的中断
10.     IER &= M_INT1;
11.                  // 在 PIEIER 寄存器层级，使能 XINT1 对应中断，并屏蔽该组其它中断
12.   // 注意：初始化中已经将 PieCtrlRegs.PIEIER1.bit.INTx4 = 1;
13.     PieCtrlRegs.PIEIER1.all &= 0x40;
14.                 // TINT0 通过 PIE 向 CPU 发出中断请求后，硬件自动将 ACK1=1;
15.   // 而 XINT1 也位于 Group 1 组内，需要再次打开 ACK 位
16.     PieCtrlRegs.PIEACK.all = 0xFFFF;
17.   // 等待 PIE 和 CPU 中断通道准备就绪，用于防止误操作
18.     asm("     NOP");
19.                  // 进入 timer0_isr 函数后，硬件自动关闭了中断总开关
20.   // 此处打开中断总开关
21.      EINT;
22.    /*插入 timer0 中断代码，此处可被打断*/
23.   /*    中断代码    */
24.                 // 关闭中断总开关，此后不可被嵌套，这是嵌套前的状态
25.   DINT;
26.                 // 恢复嵌套前 PIEIER 寄存器的状态
27.   PieCtrlRegs.PIEIER1.all = TemPIEIER;
28.                 // 清除 timer0 中断标志位，以便下次触发中断
```

```
29.     Cputimer0Regs.TCR.bit.TIF = 1;
30.
31.     // 恢复返回main 函数的地址以及其它状态，必须在DINT 后执行
32.     _RESTORE_IRQ_STATUS();
33.     }
```

3.4 IQMath 库

中科昊芯提供了与友商一样的 IQMath 库文件，HX2000 芯片的工作频率（120MHz）是友商同款产品（60MHz）的 2 倍，算法性能较友商芯片有 50%~110%的提升。

当然实际的性能还和客户的实际代码的规范性也有一定的关系。



使用方式与友商一样，可以直接调用。

```
72      _iq8 aa,bb;
73      aa = _IQ8(0.345);
74      bb = _IQ8sin(aa);
75
```

如果有需要，可以联系中科昊芯获取 IQMath 函数源码。

还有一个 IQsinTable （或者 _IQsinTable），在引入 IQmathLib.h 之后 我们也是可以直接使用的。

3.5 RPT 指令

在友商的指令集中，有 RPT 指令，一般会用来进行延时函数的实现，比如

```
asm(" RPT #5 || NOP ");
```

在中科昊芯的 HX2000 系列芯片中，也提供了实现同样功能的指令，实现方式如下：

```
asm volatile (".align 2;RPTI 5,4;NOP");
```

3.6 ENPIE

友商有一个 PIE 的使能位，使用的时候，需要提前配置，如

```
PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
```

中科昊芯的 DSC 系列产品，**默认使能 PIE 模块，无需配置，如果有，需注释掉**。

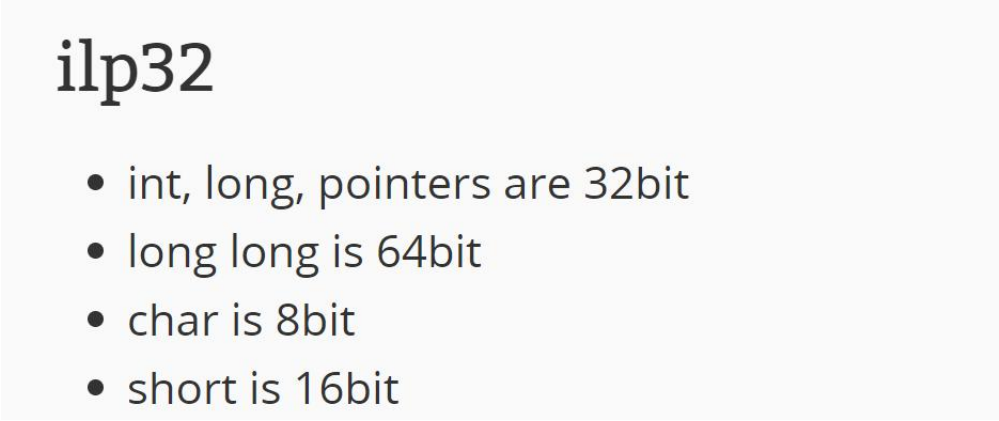
这一版本，我们把 PieCtrlRegs.PIECTRL.bit.ENPIE 这个位加上了，但是其实是没有意义的。

3.7 Uint 类型

友商 2000 系列芯片是 32 位 DSP，中科昊芯 HX2000 系列芯片是 32 位 RISC-V DSP，对 int 类型的解释不一样，使用中需要注意。

对于之前友商程序中有定义 int 类型的数组或者变量，可以手动改成 int16 来降低存储空间占用。

RISC-V 32 位处理器支持的数据类型如下：



中科昊芯 Haawking IDE 建议使用类型+位长的数据类型格式：

```
1.    typedef  unsigned char  Uint8;
2.    typedef  unsigned short int  Uint16;
3.    typedef  unsigned int  Uint32;
```

```
typedef unsigned long long int Uint64;
typedef  signed char  Int8;
typedef  signed short int Int16;
typedef  signed int Int32;
typedef signed long long int Int64;
typedef  unsigned char  uint8;
typedef  unsigned short int  uint16;
typedef  unsigned int  uint32;
typedef unsigned long long int uint64;
typedef  signed char  int8;
typedef  signed short int int16;
typedef  signed int int32;
typedef signed long long int int64;
```

上述定义，已经在中科昊芯提供的驱动文件中（DSP2803x_Device.h，F2802x_Device.h），完全兼容友商的定义，用户无需重新定义；作为对比，下方列出了友商的定义（DSP2803x_Device.h，F2802x_Device.h）。

```
typedef int                 int16;
typedef long                int32;
typedef long long           int64;
typedef unsigned int        Uint16;
typedef unsigned long       Uint32;
typedef unsigned long long  Uint64;
typedef float               float32;
typedef long double         float64;
```

在跟客户交流的过程中，我们发现有客户重新定义

```
typedef long long     llong;
typedef unsigned int   Uint;
typedef unsigned long   Ulong;
typedef unsigned long long   Ullong;
```

因此，我们建议客户在使用的时候，对不同位宽的数据类型特别注意。

3.8 因为数据类型导致的一些特殊的处理

3.8.1 sizeof()使用（基于 Haawking 8 位寻址）

sizeof()用于返回一个变量或者类型的大小， 返回的结果是一个整数，在 Haawking IDE 中，sizeof()返回结果的单位为 1 个字节；但在 CCS 中，sizeof()返回的结果的单位是 2 个字节，这是由于编译器不同导致。
因此， Haawking IDE 中 sizeof()返回值的结果等于 CCS 中 sizeof()返回值结果 * 2；

例如：在 CCS 中： `int x = sizeof(uint16_val);`

Haawking IDE 中：`int x = sizeof(uint16_val)/2;`

3.8.2 memcpy(void* dest, const void* src, size_t len)的使用 （基于 Haawking 8 位寻址）

在 CCS 中，memcpy（）是按每个元素 16 位拷贝，在 Hawwking IDE 中，是按每个元素 8 位拷贝；

在 CCS 中：

```
memcpy(&gSendToMotor2MsDataBuff[100],&funcCode.code.PosSetAdd,8);
```

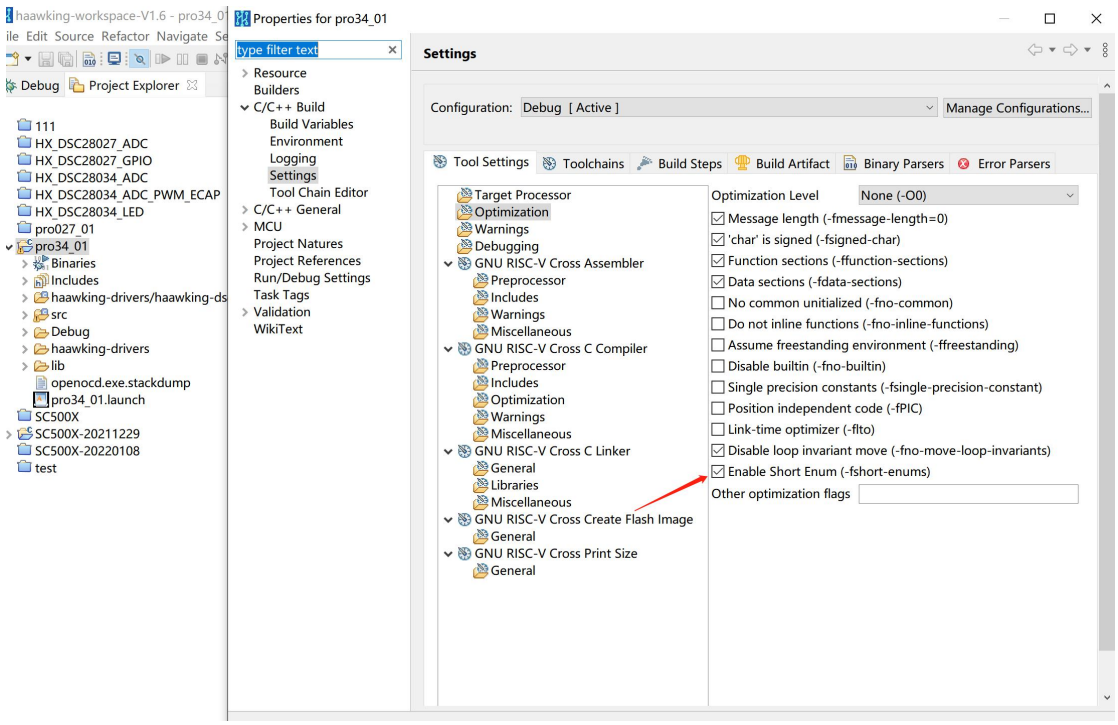
在 Haawking IDE 中：

```
memcpy(&gSendToMotor2MsDataBuff[100], &funcCode.code.PosSetAdd, (8*2));
```

3.8.3 typedef enum 使用（基于 Haawking 8 位寻址）

CCS 中一个枚举类型数据默认占用 2 个字节， 在 Haawking IDE 中，一个枚举类型数据默认占用 4 个字节。

当需要两个字节的枚举的时候，首先需要修改结构体的定义， 然后勾选优化面板中的 -fshort-enums。如下图：



例如：

在 CCS 中

```
typedef enum CONTROL_MOTOR_TYPE_ENUM_DEF{
    ASYNC_SVC,
    ASYNC_FVC,
    ASYNC_VF,
 SYNC_SVC = 10,
    SYNC_FVC,
    SYNC_VF,
 DC_CONTROL = 20,
    RUN_SYNC_TUNE
}CONTROL_MOTOR_TYPE_ENUM;
```

在 Hawwking IDE 中

```
typedef enum CONTROL_MOTOR_TYPE_ENUM_DEF{
    ASYNC_SVC,
    ASYNC_FVC,
    ASYNC_VF,
 SYNC_SVC = 10,
    SYNC_FVC,
    SYNC_VF,
 DC_CONTROL = 20,
 DC_CONTROL_Hx = 1000,
    RUN_SYNC_TUNE
}CONTROL_MOTOR_TYPE_ENUM;
```

以上根据实际需要来进行修改；

同时，因为中科昊芯 HX2000 系列 DSP 芯片，支持 8 位数据类型，如果有些变量或者数组的取值范围在 Char 类型的表示范围之内，可以减少程序代码量